MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

ECE3136 – JAVA TECHNOLOGY

(All Sections / Groups)

15 OCTOBER 2016
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of FIVE (5) pages including cover with THREE (3) questions only.
2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.
3. This examination is OPEN-BOOK. Allowed material: Textbook and/or bound notes only.
4. Write answers neatly in the Answer Booklet provided.

Question 1

(a) Create a program, `ArithmeticChallenge`, that poses a series of arithmetic questions to the user (to answer), then computes the number of correctly answered ones, and the average time (in seconds) taken per question. Display either an addition or subtraction problem. The operands are randomly-generated values of 1-100, inclusive. The operation (addition or subtraction) is also determined randomly.

[10 marks]

Obtain the system time right after the question is displayed and again right after an input is read. The difference is the amount of time taken by the user to answer the question.

[3 marks]

After each question, prompt the user to choose whether to answer another (y/n?). Repeat for as long as user inputs "y". See Figure Q1a.

[2 marks]

Output: display number of questions answered correctly and average time taken per question in a dialog box. See Figure Q1b.

[6 marks]

Handle all exceptions.

[4 marks]

```
Arithmetic Challenge!
30-39=-9
Another? (y/n): y
48+16=56
Another? (y/n): y
68-78=-10
Another? (y/n): n
```

Figure Q1a: Sample Output (Command Window)

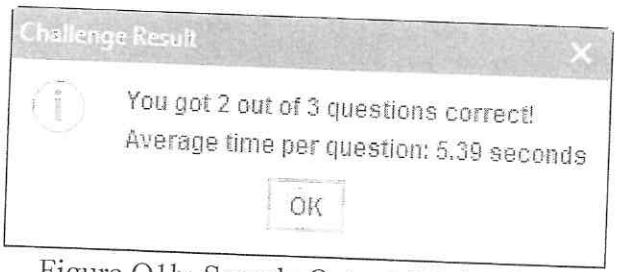

Figure Q1b: Sample Output (Dialog box)

Hint: method `System.currentTimeMillis()` returns current system time in milliseconds

Question 2

Create a program to allow users to calculate the monthly repayments for a car loan. Write two classes: `LoanParameters` and `LoanCalculator`.

(a) `LoanParameters`

Create a class, `LoanParameters`, which models a set of parameters in a car loan. It has six instance variables: a `String` which represents the loan's description, a `double` which represents the car price, a `double` which represents the down payment, a `double` which represents the loan term, and a `double` which represents the interest rate in percent and a `double` which represents the monthly repayment.

[2 marks]

Provide a constructor and two mutator methods. The constructor takes in six arguments (corresponding to the instance variables) and passes them to the mutator methods. The first mutator method sets the loan description string according to the format "<Price>/<Down Payment>/<Term>/<Interest Rate>" and the second mutator method calculates the monthly repayment.
Monthly repayment formula:

$$Monthlyrepayment = \frac{(price - DownPayment) \times \left(1 + \frac{(Term \times InterestRate}{100}\right)}{Term \times 12}$$

[5 marks]

Provide a `toString` method which returns the loan's description and monthly repayment, separated by a tab character.

[1 mark]

(b) `LoanCalculator`

Create a class, `LoanCalculator`, which inherits from `JPanel`. Include user interface (UI) elements of four labels, four text fields, one button and an area field. Attach a listener to the button. Add each element to the panel. Refer to Figure Q2.

[10 marks]

After each item is entered in the text field and the button is pressed, the values will be used to create an object of class `LoanParameters` created in part (a). Handle all exceptions that may occur in obtaining user inputs by printing a message in the command prompt.

[5 marks]

Use the `LoanParameters` object created to assist in displaying the newly-entered item as output in the text area. Each new item must be appended to the existing list.

[2 marks]

NOTE: DO NOT write the driver program.

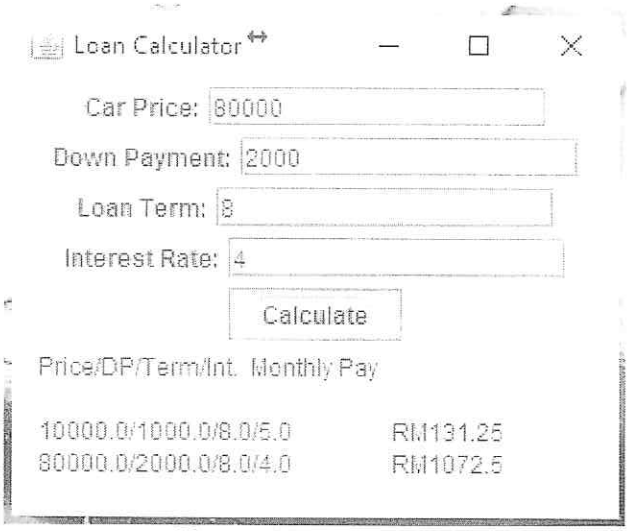

Figure Q2: Loan Calculator

Question 3

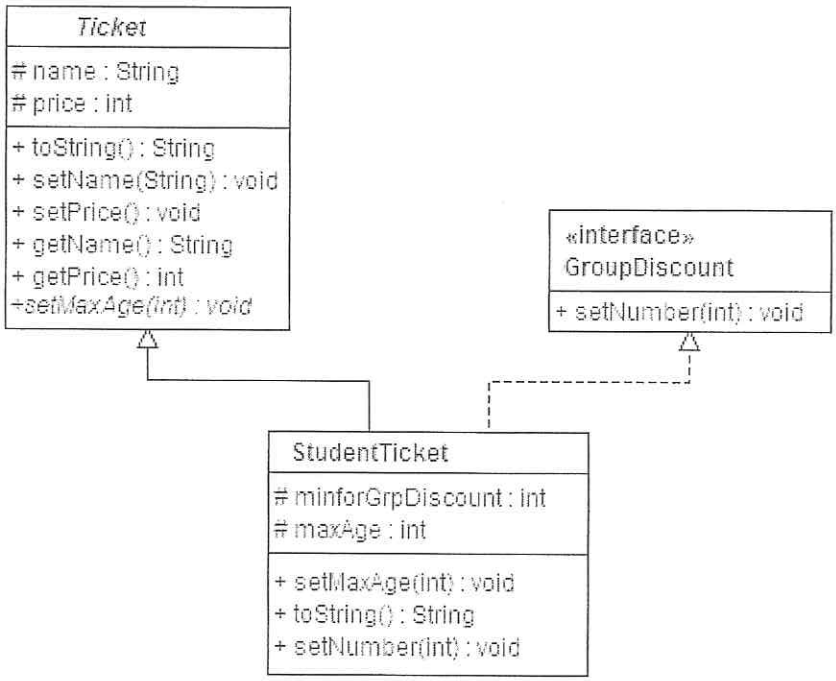

Figure Q3

Continued...

Refer to the UML diagram in Figure Q3. In your programs, include all class members shown in the UML diagram.

(a) Create abstract class `Ticket`.
Provide mutator and accessor methods. The mutator must only allow the value of `price` to be set to a number ranging from 0 to 110. Output an error message otherwise. Provide a constructor, a `toString` method and the abstract method `setMaxAge`.

[11 marks]

(b) Create interface `GroupDiscount`.

[2 marks]

(c) Create a class `StudentTicket`.
This class inherits from `Ticket` and implements interface `GroupDiscount`. Override the superclass `toString` method.

[3 marks]

Provide a constructor which uses the superclass constructor. It takes in the arguments for name of ticket, price, maximum age and minimum size for group discount.

[2 marks]

Override abstract method `setMaxAge` which sets the maximum age for the student ticket eg. for kindergarten students ticket, the maximum age is 6. The method `setMaxAge` should only allow the age to be set if the value is between 6 and 26. Output an error message otherwise.

[2 marks]

Override abstract method `setNumber` to set the minimum size for group discount. The minimum size for a group cannot be negative number.

[2 marks]

(d) Create a simple driver class to test `StudentTicket` class. The driver program must print out all the instance variables' values. It should also include cases with erroneous arguments to check error detection.

[3 marks]

End of Paper